Trey McAtee

10/4/20

CS 350 Assignment 1

**as1Q1a.c**

**Problem Definition/Specification**

We need to write a function that will take a base number and exponentiate it to a set number of times. The program will then output the answer. We must use a FOR loop to control this calculation, and we are assuming that exponent is a positive and nonzero integer, while base is an integer. This problem does not specify the need for user input. We are not able to use any math library functions.

**Problem Analysis**

We have two inputs, the base and the exponent. There is one output, the answer to the exponentiation.

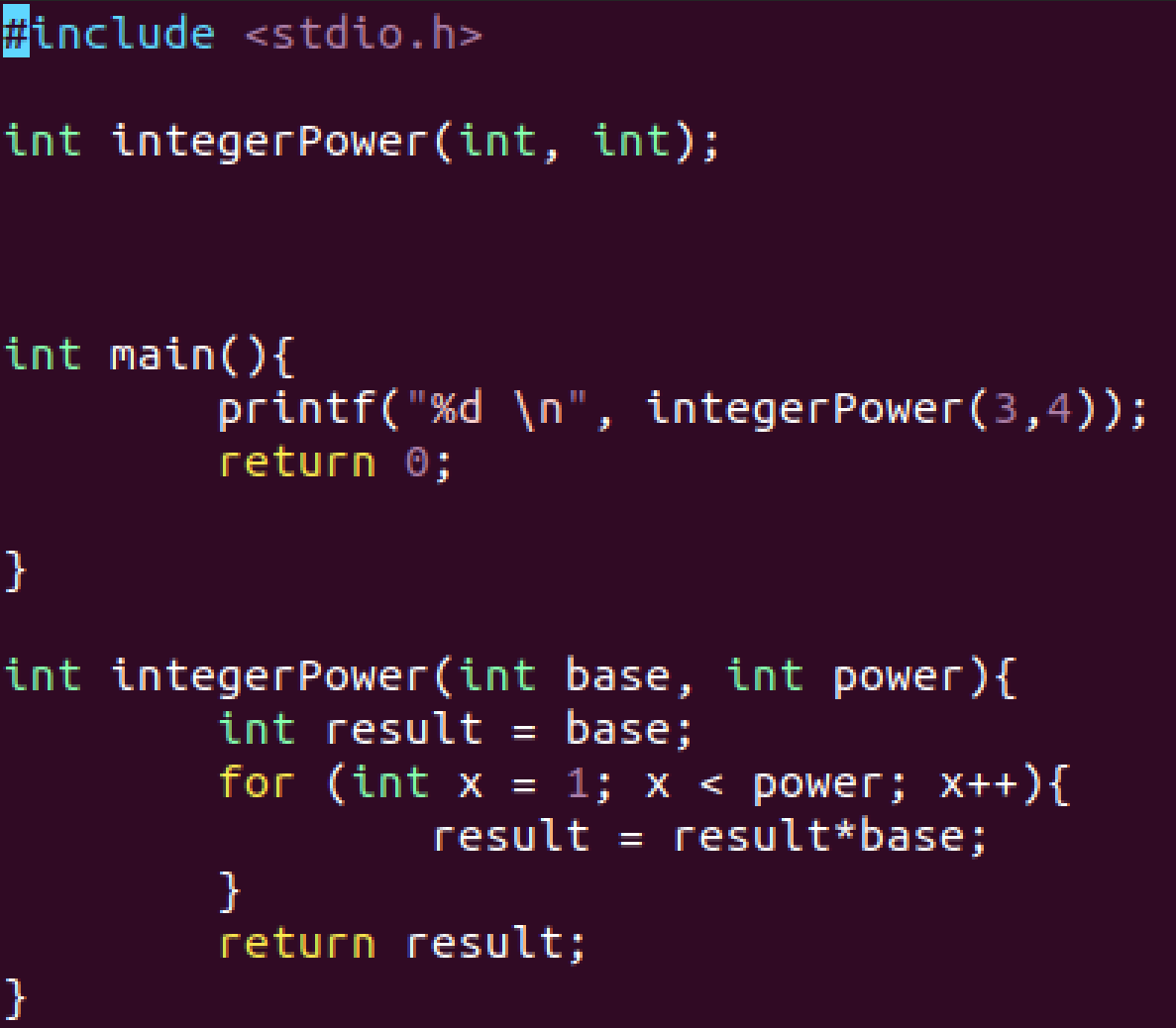
**Algorithm Design**

In main, we will print out the output to the created function, integerPower. The function integerPower will take two positive, nonzero integers with the first being the base and the second being the exponent. There will be an integer variable to store the results. Next, there will be a for loop to control how many times the operation is performed. Finally, the result variable will be returned as the output to the function.

**Test Plan**

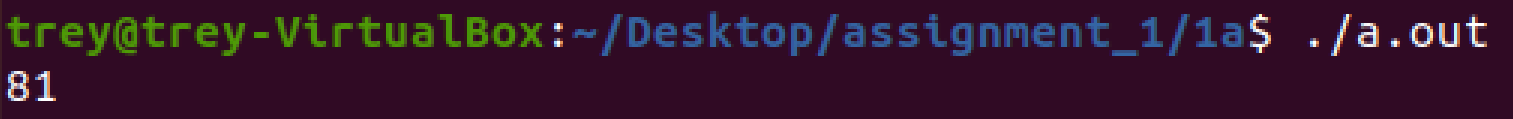
This problem does not require user input, so testing the program will consist of manually editing the values the function integerPower receives when it is called in main(). I will use the value pairs of (3,4), (2,5), (5,6), (4,3), and (5,4) in that order to test the program, and verify the results using a calculator.

**Implementation**

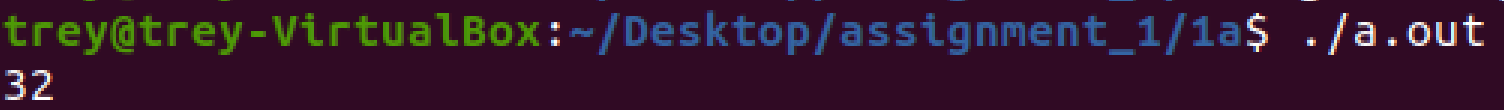
****

**Testing**

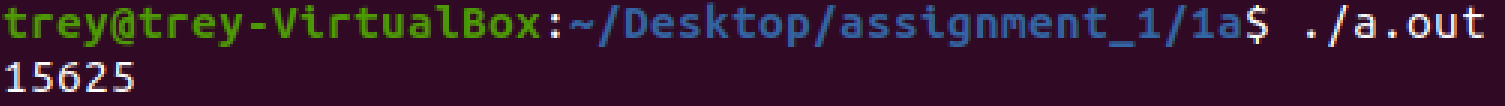
Test 1 (3,4) :



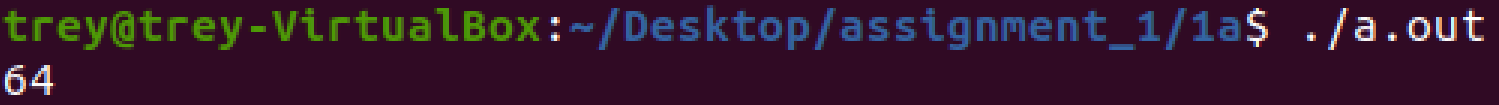
Test 2 (2,5) :



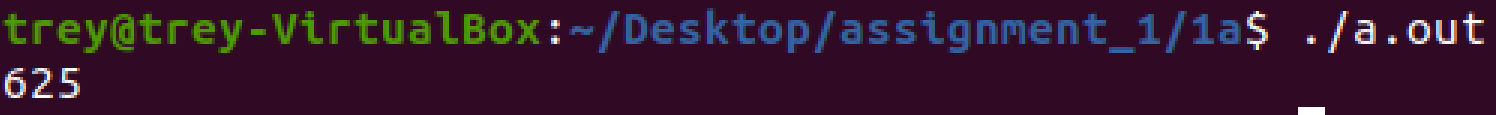
Test 3 (5,6) :



Test 4 (4,3) :



Test 5 (5,4) :



**as1Q1b.c**

**Problem Definition/Specification**

We need to write a function that will take a base number and exponentiate it to a set number of times. The program will then output the answer. We must use recursion to complete this task.

**Problem Analysis**

We have two inputs, the base and the exponent. There is one output, the answer to the exponentiation.

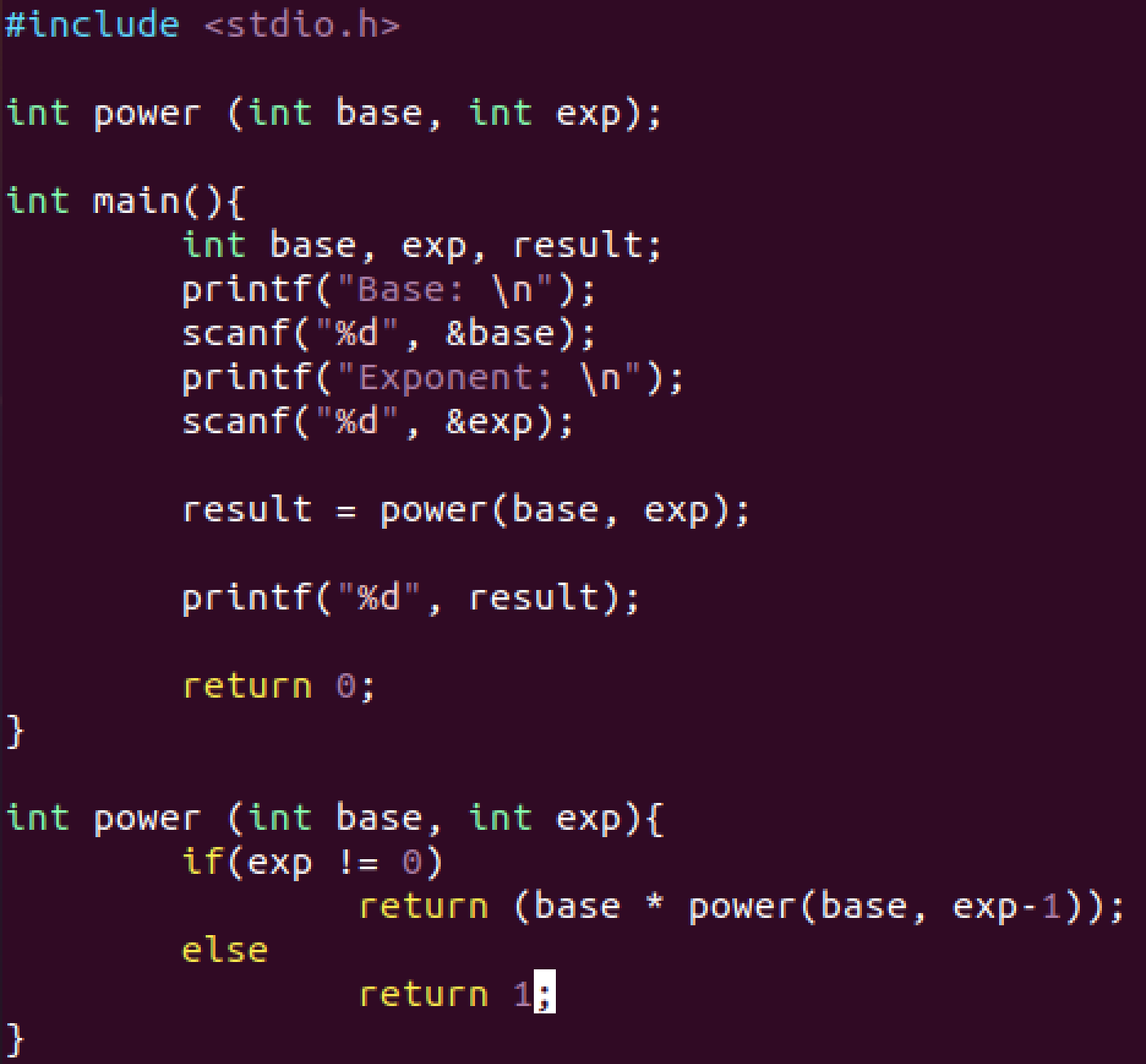
**Algorithm Design**

In main, we will print out the output to the created function, power. The function power will take two positive, nonzero integers with the first being the base and the second being the exponent. There will be an integer variable to store the results. When doing this recursively, I can have base multiplied by the function call with the exponent subtracted by 1. To accomplish the task in this manner, however, I will need to check that the operation is not performed when the exponent becomes 0.

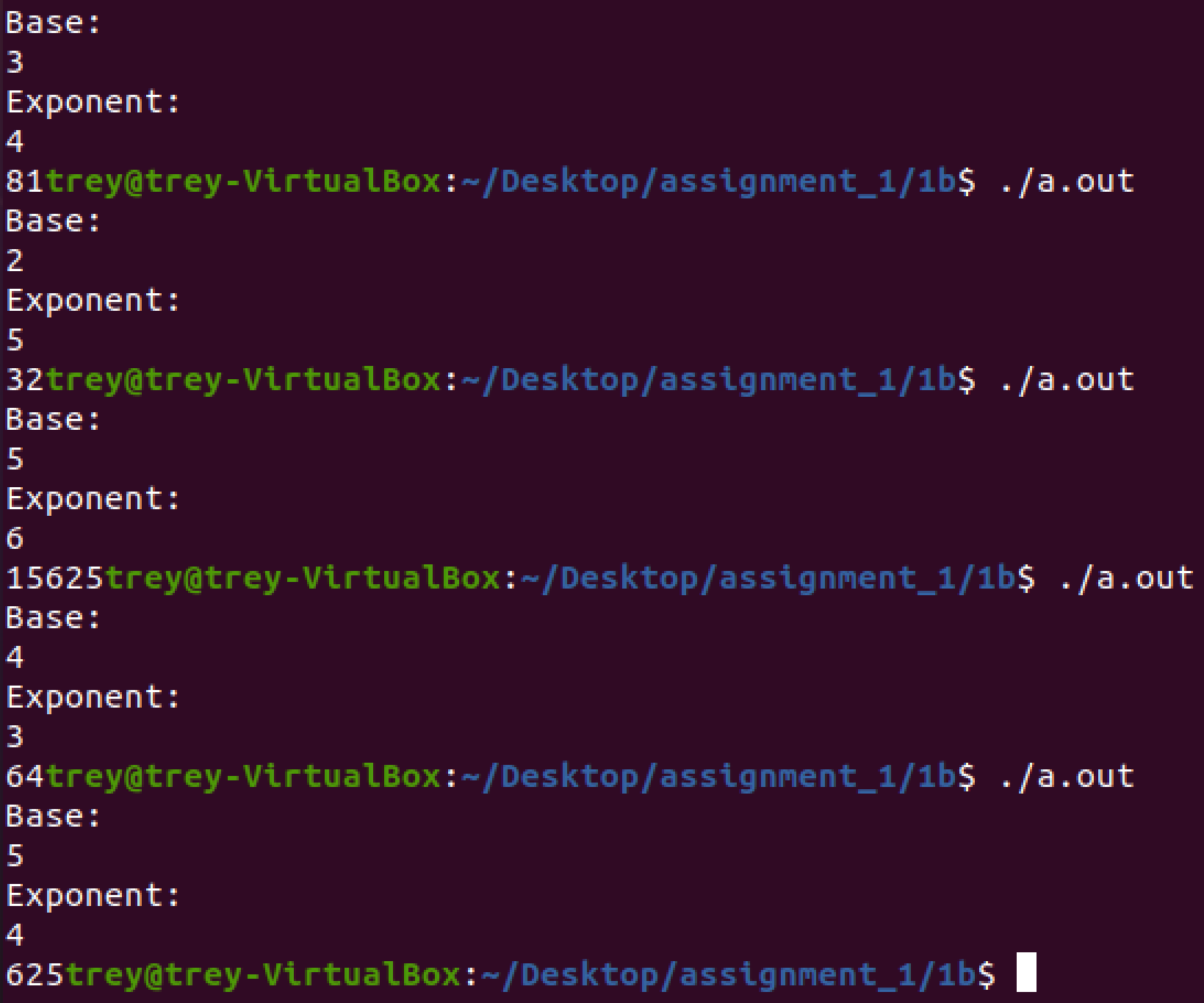
**Test Plan**

I will use the value pairs of (3,4), (2,5), (5,6), (4,3), and (5,4) in that order to test the program, and verify the results with those of my results from question 1a.

**Implementation**



**Testing**



**as1Q2a1.c**

**Problem Definition/Specification**

We need to write a function that will take a number and determine whether or not it is a prime number.

**Problem Analysis**

We have one input, the number the user gives to determine whether or not it is prime. There is also one output, whether or not the inputted number was prime.

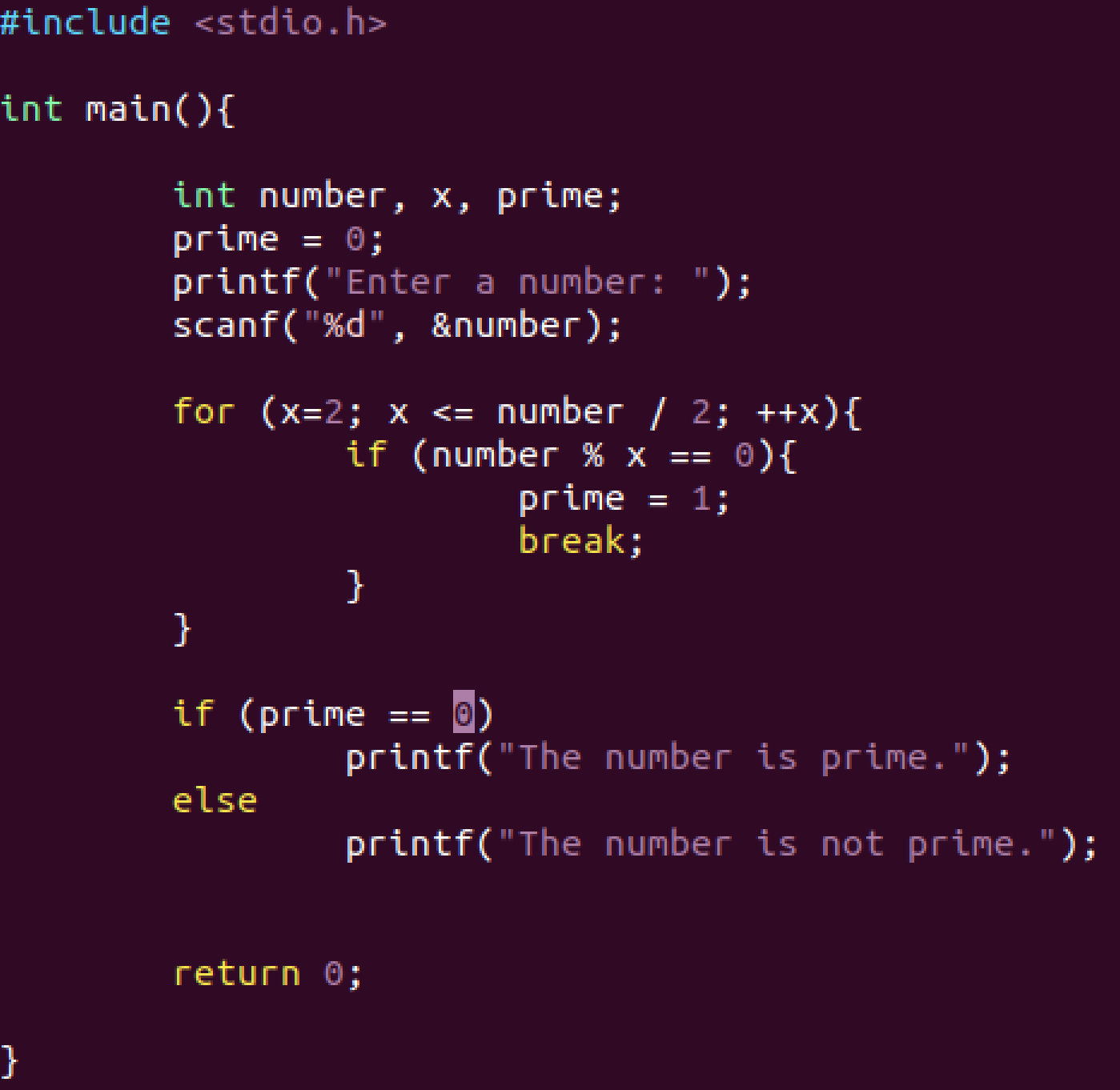
**Algorithm Design**

In main, we will have 3 integer variables. One will store the users input, one will be the controlling variable in a for loop, and one will change from 0 to 1 based on whether the number is prime. I will have the loop control variable = 2 at start, because prime numbers should only be divisible by one and themselves. As soon as the number is divisible by something other than one or itself, the loop will break and the program should say that the number is not prime.

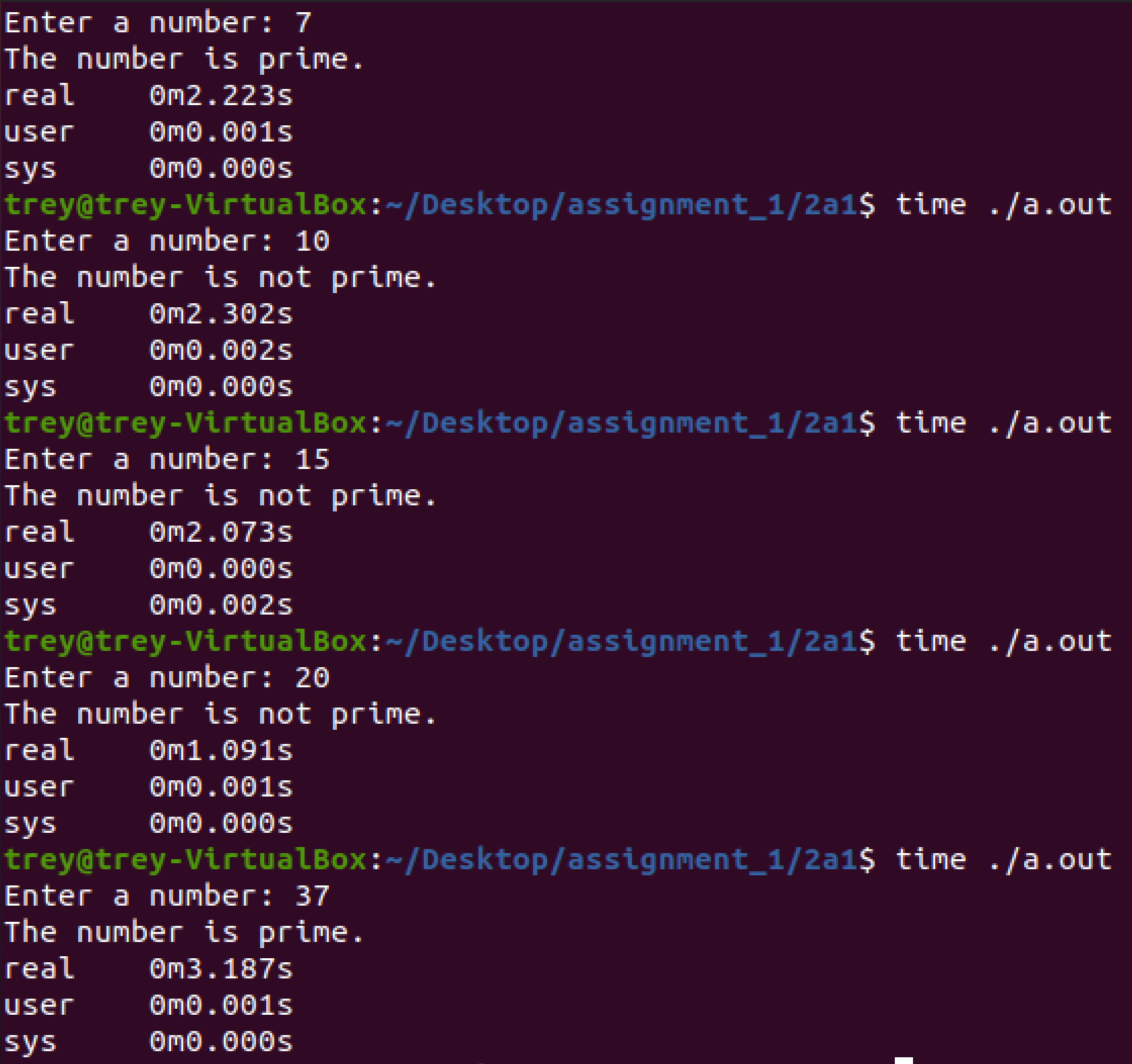
**Test Plan**

I will use the values of 7, 10, 15, 20, and 37 to test the program, in that order. These are chosen somewhat randomly, as I made sure to include at least 2 prime numbers. I will also call the time function to adhere to the extra request on the homework, which will be discussed later on.

**Implementation**

****

**Testing**



**as1Q2a2.c**

**Problem Definition/Specification**

We need to write a function that will print out every prime number between one and 10000, and then be able to test certain prime numbers still.

**Problem Analysis**

We have one input, the number the user gives to determine whether it is prime. There are also multiple outputs, whether the inputted number was prime and every prime number from one to 10000.

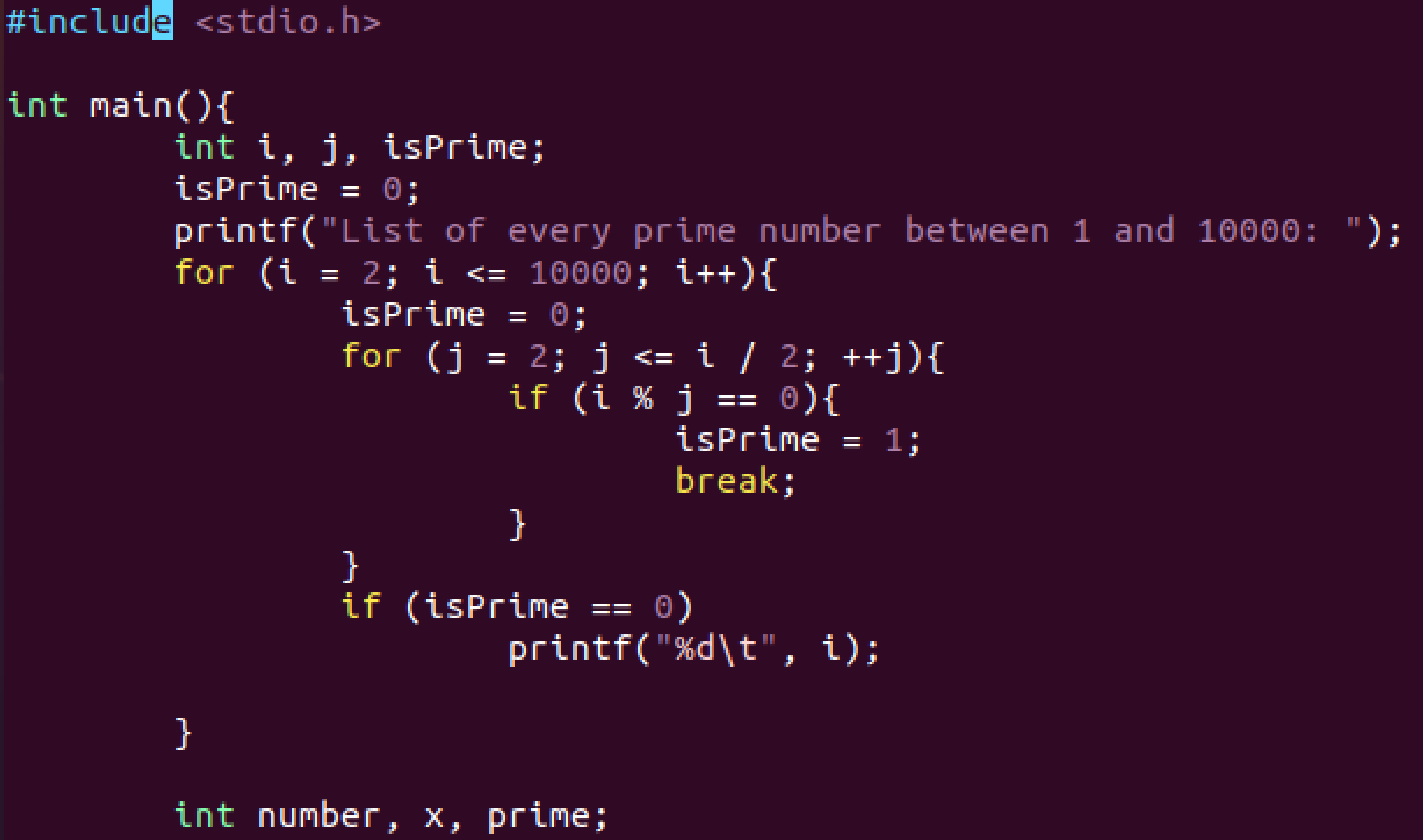
**Algorithm Design**

In main, we will have 6 integer variables. One will store the users input, 3 will be the controlling variable in a for loop, and two will change from 0 to 1 based on whether the number is prime. I will have the loop control variables = 2 at start, because prime numbers should only be divisible by one and themselves. The program will print out every prime number from 1 to 10000 before asking for user input. When testing certain numbers, as soon as the number is divisible by something other than one or itself, the loop will break, and the program should say that the number is not prime.

**Test Plan**

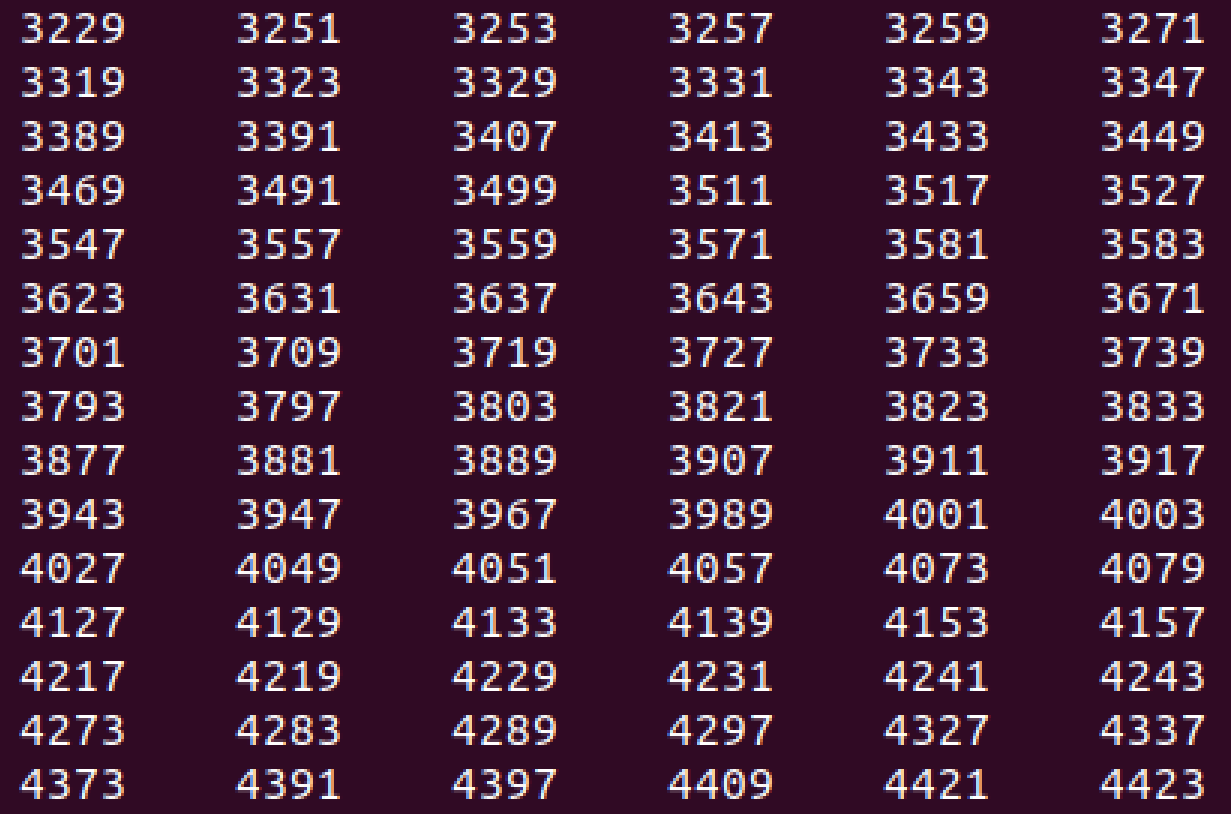
I will pick some numbers that my program prints out and compare them to a list of prime numbers to confirm whether every number it prints is indeed a prime number. I will also call the time function to adhere to the extra request on the homework, which will be discussed later.

**Implementation**

****

The program beyond the last line is the same as in q2a1.

**Testing**



I have compared most of these numbers to a list of prime numbers and also within my own program, as it contains functionality to test if a number is prime.

Part C Answer:

There isn’t a clear performance increase to the human eye, but the time function appears to show some improvement in efficiency.

**as1Q2b.c**

**Problem Definition/Specification**

We need to program the sieve of Eratosthenes problem, which is a way to find prime numbers.

**Problem Analysis**

We have one input, the number the user gives to determine the array length of how far up to go. There are many outputs, the full list of prime numbers up to the chosen number.

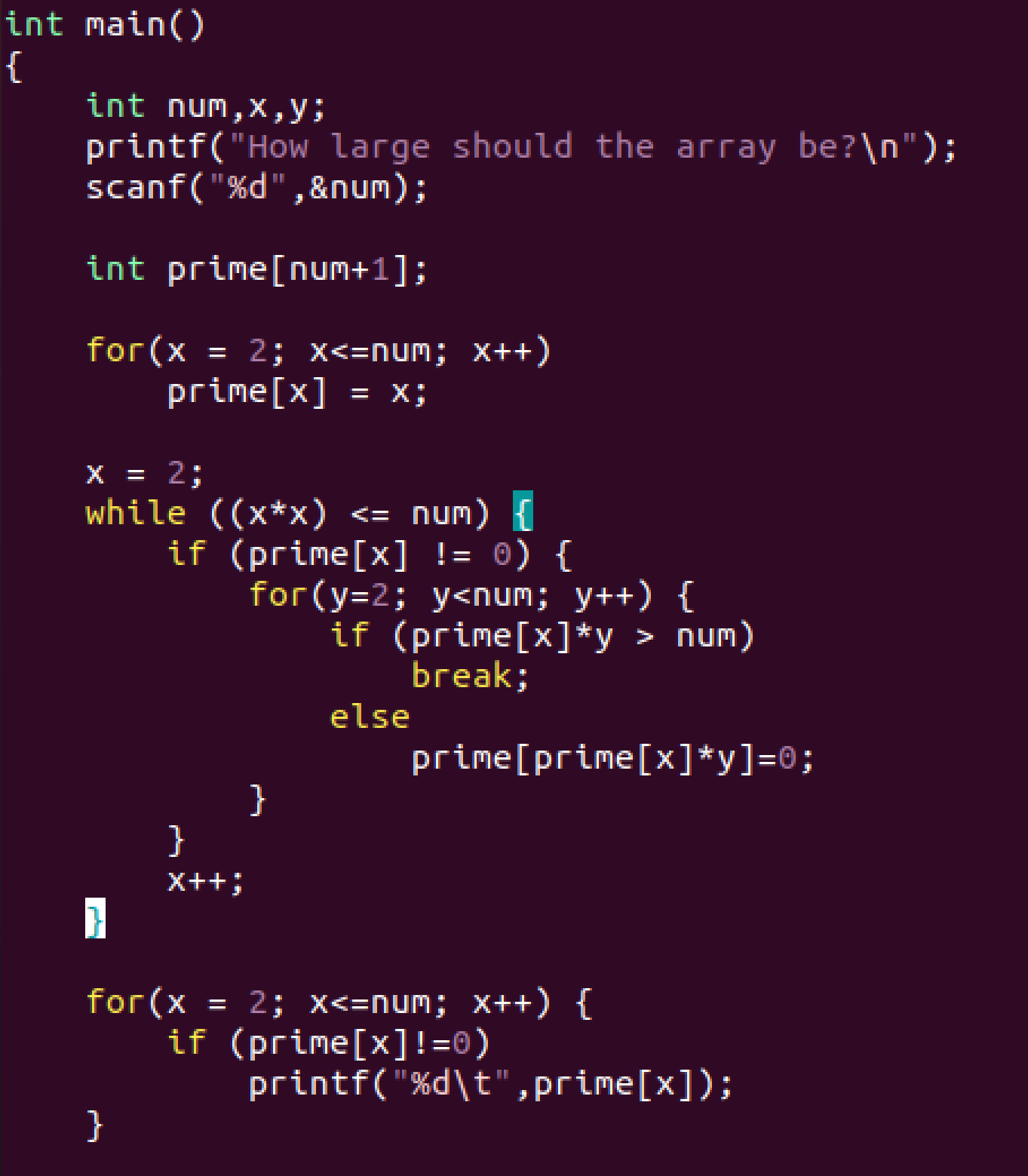
**Algorithm Design**

In main, I will create an array based on user input and use looping techniques to find every prime number from 2 to the number the user entered.

**Test Plan**

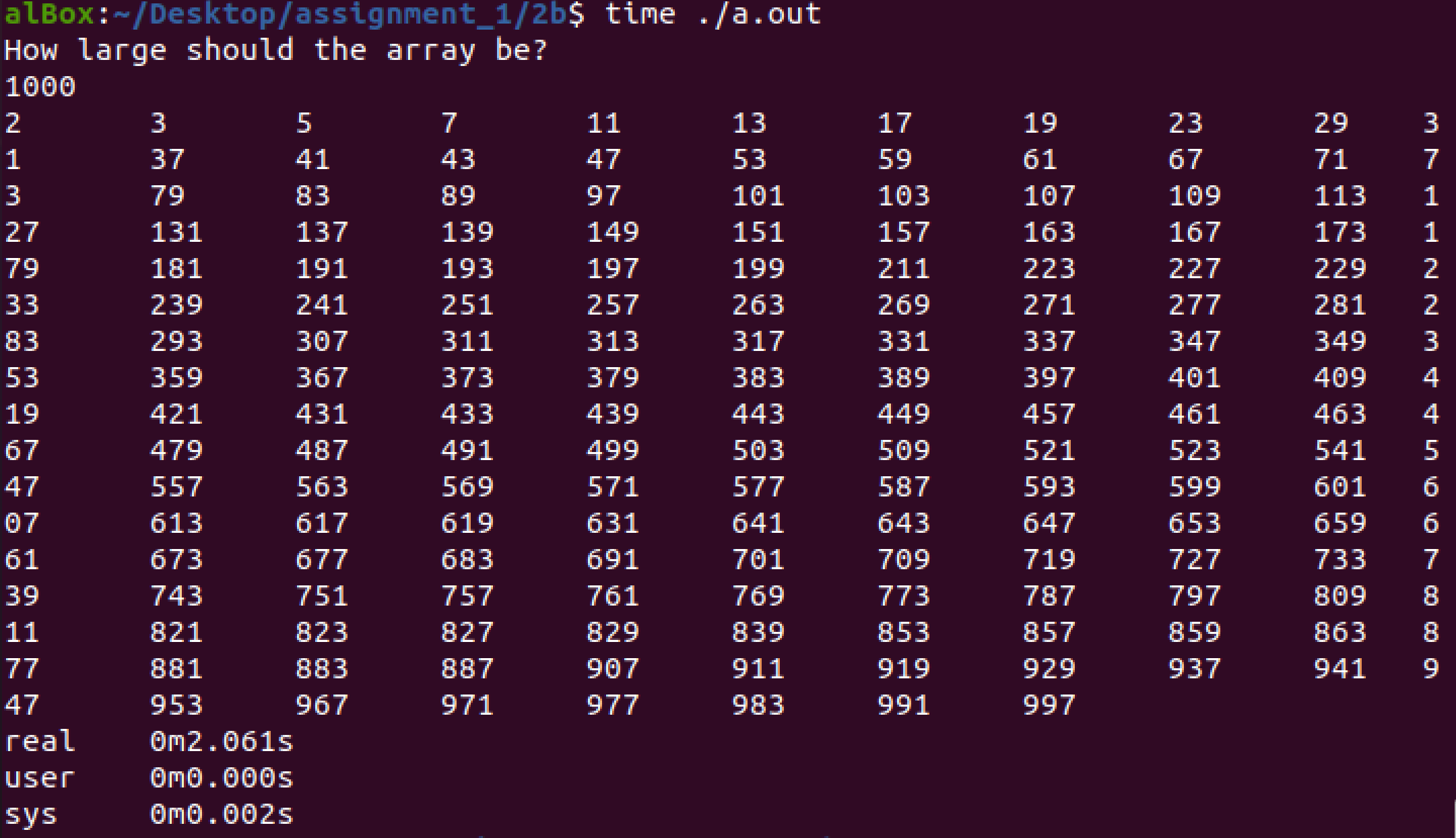
I will simply run the program and compare the numbers it shows with a table of prime numbers to see if it is correct.

**Implementation**



The program beyond the last line is just return 0 to end the main method.

**Testing**



**as1Q3.c**

**Problem Definition/Specification**

We need to program the Turtle Graphics problem, which is essentially a way to draw things in an array floor with c. I need to keep track of all commands, and execute them sequentially once the program is meant to be run.

**Problem Analysis**

We have one input, the number the user gives to determine what command to execute. Some commands that must be implemented are printing, ending the program, putting the pen up or down, turning right and left, and moving forward a set number of spaces.

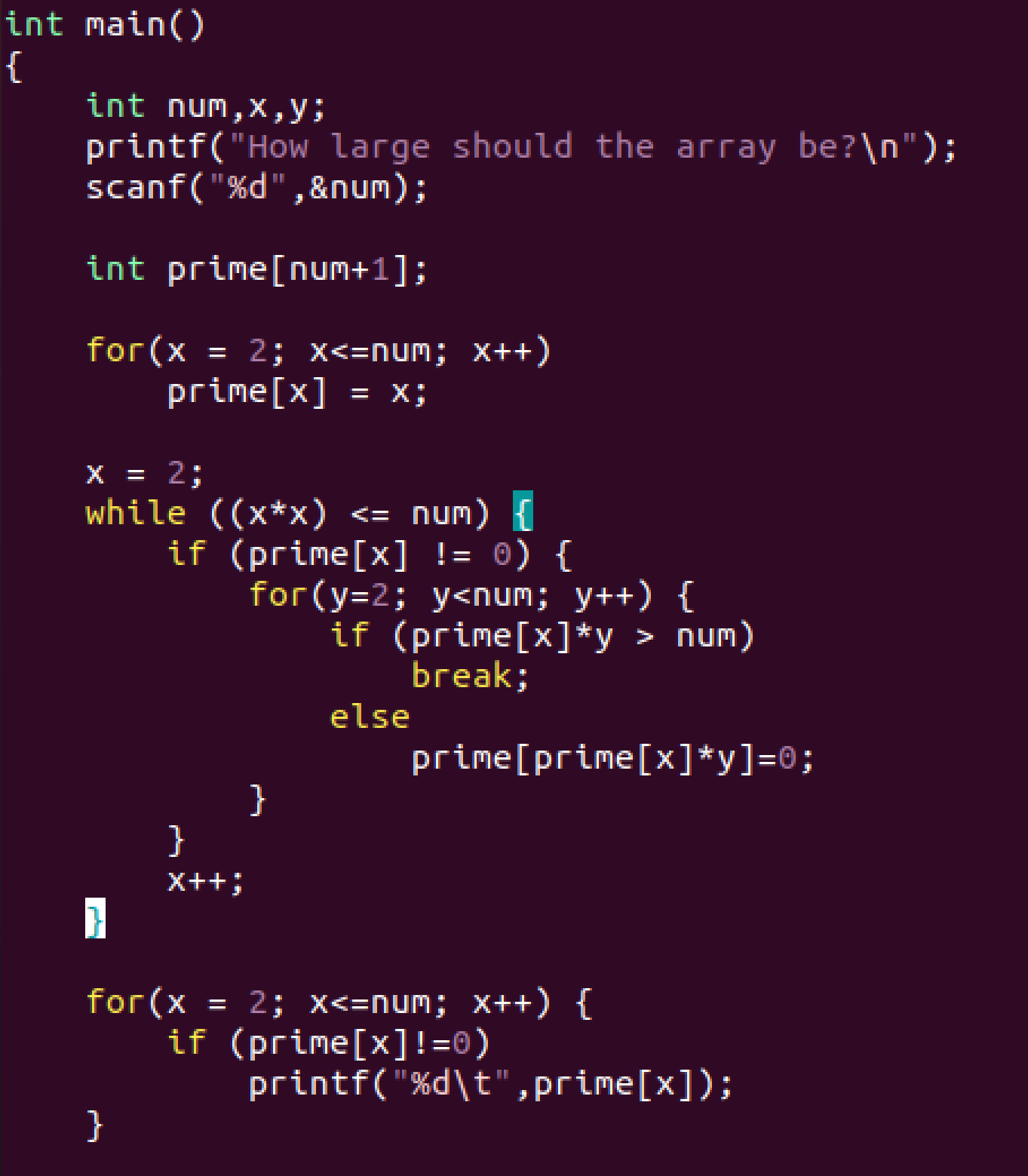
**Algorithm Design**

I will use a variety of for loops and a switch statement to complete my objective, while declaring variables and possibly constants as early as possible to allow the code to be as legible as possible, as I believe this will be a long program. (\*Unable to create constants, brings errors at every attempt\*)

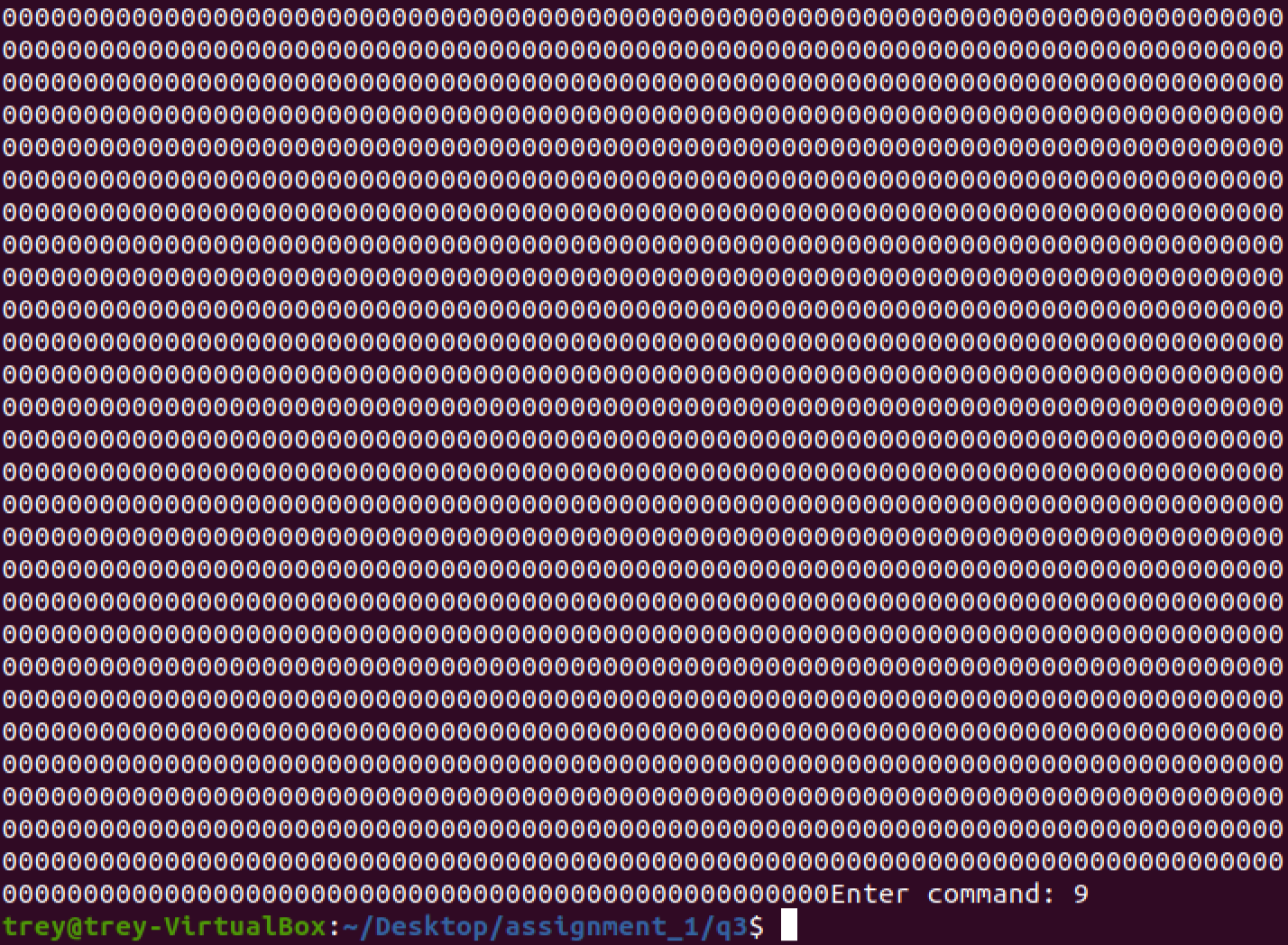
**Test Plan**

I will simply run the program with the command list given in the problem in the book and see the output that it gives. It should line up with the description of the example command list from the book.

**Implementation**



**Testing**



I was unable to get working results on this problem, as no matter what I seemed to do the program would either simply crash or print out all 0’s. It does take commands correctly, however.